

| | | | | | | |
|---------------|-----|------------|---------------|------------|------------|-----------------|
| SSSSSSSSSSSSS | YYY | YYY | SSSSSSSSSSSSS | LLL | 0000000000 | AAAAAAA |
| SSSSSSSSSSSSS | YYY | YYY | SSSSSSSSSSSSS | LLL | 0000000000 | AAAAAAA |
| SSSSSSSSSSSSS | YYY | YYY | SSSSSSSSSSSSS | LLL | 0000000000 | AAAAAAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAA AAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAA AAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAA AAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAA AAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAA AAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAA AAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAA AAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAA AAA |
| SSSSSSSSSS | YYY | YYY | SSSSSSSSSS | LLL | 000 | 000 AAA AAA |
| SSSSSSSSSS | YYY | YYY | SSSSSSSSSS | LLL | 000 | 000 AAA AAA |
| SSSSSSSSSS | YYY | YYY | SSSSSSSSSS | LLL | 000 | 000 AAA AAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAAA AAAAAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAAA AAAAAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAAA AAAAAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAA AAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAA AAA |
| SSS | YYY | YYY | SSS | LLL | 000 | 000 AAA AAA |
| SSSSSSSSSS | YYY | SSSSSSSSSS | LLLLLLLLLLLL | 0000000000 | AAA | AAA |
| SSSSSSSSSS | YYY | SSSSSSSSSS | LLLLLLLLLLLL | 0000000000 | AAA | AAA |
| SSSSSSSSSS | YYY | SSSSSSSSSS | LLLLLLLLLLLL | 0000000000 | AAA | AAA |

_S2
Syn

SS1
SS1
SS1
SS1
SS1
SS1
SS1
SYS
SYS
SYS
TRY
UNL
WR1

FILEID**CSPMOUNT

14

CCCCCCCC SSSSSSSS PPPPPPPP MM MM 000000 UU UU NN NN NN TTTTTTTT
CCCCCCCC SSSSSSSS PPPPPPPP MM MM 000000 UU UU NN NN NN TT
CC SS PP PP MMMM MMMM 00 00 UU UU NN NN NN TT
CC SS PP PP MMMM MMMM 00 00 UU UU NNNN NN NN TT
CC SS PP PP MM MM 00 00 UU UU NNNN NN NN TT
CC SSSSSS PPPPPPPP MM MM 00 00 UU UU NN NN NN TT
CC SSSSSS PPPPPPPP MM MM 00 00 UU UU NN NN NN TT
CC SS PP MM MM 00 00 UU UU NN NN NNNN TT
CC SS PP MM MM 00 00 UU UU NN NN NNNN TT
CC SS PP MM MM 00 00 UU UU NN NN NN TT
CCCCCCCC SSSSSSSS PP MM MM 000000 UUUUUUUUUU NN NN NN TT
CCCCCCCC SSSSSSSS PP MM MM 000000 UUUUUUUUUU NN NN NN TT

A 10x10 grid of letters. The first column contains ten 'L's. The second column contains two 'I's followed by eight 'L's. The third column contains three 'I's followed by seven 'L's. The fourth column contains four 'I's followed by six 'L's. The fifth column contains five 'I's followed by five 'L's. The sixth column contains six 'I's followed by four 'L's. The seventh column contains seven 'I's followed by three 'L's. The eighth column contains eight 'I's followed by two 'L's. The ninth column contains nine 'I's followed by one 'L'. The tenth column contains ten 'S's.

CSPM
V04-1

1234567891011121314151617181920212223242526272829293031323334353637383939404142434445464748495051525354555657

0001 0 MODULE CSPMOUNT
0002 0 (LANGUAGE (BLISS32)
0003 0 IDENT = 'V04-000'
0004 0 ; =
0005 0 *****
0006 0 *
0007 0 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0008 0 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0009 0 * ALL RIGHTS RESERVED.
0010 0 *
0011 0 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0012 0 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0013 0 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0014 0 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0015 0 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0016 0 * TRANSFERRED.
0017 0 *
0018 0 *
0019 0 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0020 0 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0021 0 * CORPORATION.
0022 0 *
0023 0 *
0024 0 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 0 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 0 *
0027 0 *****
0028 0 *
0029 0 ++
0030 0
0031 0 FACILITY: MOUNT,CSP
0032 0
0033 0 ABSTRACT:
0034 0
0035 0 This module contains the cluster server action routine for
0036 0 MOUNT and is part of the Cluster Server Process (CSP).
0037 0
0038 0 Environment:
0039 0
0040 0 Full process context capable of kernel mode.
0041 0
0042 0 Author:
0043 0
0044 0 Hai Huang
0045 0
0046 0 Creation date:
0047 0
0048 0 28 Feb 1984
0049 0
0050 0
0051 0 Revision history:
0052 0
0053 0 V03-003 HH0022 Hai Huang 17-May-1984
0054 0 Dismiss the mount request if the device is not
0055 0 cluster-wide, or if the device is already mounted.
0056 0
0057 0 V03-002 HH0007 Hai Huang 16-Mar-1984

; Ro
; 4
; 4

```
58      0058 0 !          Add cluster-wide group-volume support.  
59      0059 0 !  
60      0060 0 !          V03-001 HH0004      Hai Huang      01-Mar-1984  
61      0061 0 !          Add cluster-wide mount support.  
62      0062 0 !--  
63      0063 0 !--  
64      0064 0 !  
65      0065 1 BEGIN           ! Start of CSPMOUNT  
66      0066 1  
67      0067 1 LIBRARY 'SYSSLIBRARY:LIB.L32' ;  
68      0068 1 REQUIRE 'LIBS:CSPDEF' ;  
69      0262 1  
70      0263 1 LINKAGE  
71      0264 1           JSB_2 = JSB (REGISTER=2) ;  
72      0265 1  
73      0266 1 FORWARD ROUTINE  
74      0267 1  
75      0268 1           CSP$MOUNT : JSB 2,  
76      0269 1           CSP_MOUNT_DECIPHER : NOVALUE,  
77      0270 1           CSP-DISMOUNT_DECIPHER : NOVALUE,  
78      0271 1           GET_UIC,  
79      0272 1           SET_UIC,  
80      0273 1           CHECK_DEVICE;  
81      0274 1  
82      0275 1
```

```
84      0276 1
85      0277 1 %SBTTL 'CSP$MOUNT
86      0278 1 GLOBAL ROUTINE CSP$MOUNT - MOUNT client support for CSP'
87      0279 1
88      0280 1 (CSD : REF BLOCK [,BYTE]) : JSB_2 =
89      0281 1 +
90      0282 1
91      0283 1 FUNCTIONAL DESCRIPTION:
92      0284 1
93      0285 1 This routine performs the CSP mount client action routine.
94      0286 1 The possible actions are mount and dismount, depending on
95      0287 1 the parameter specified in the CSD packet.
96      0288 1
97      0289 1 INPUTS:
98      0290 1
99      0291 1 CSD : Pointer to the address of the received CSD
100     0292 1
101     0293 1 OUTPUTS:
102     0294 1
103     0295 1 None.
104     0296 1
105     0297 1 IMPLICIT INPUTS:
106     0298 1
107     0299 1 None.
108     0300 1
109     0301 1 OUTPUT PARAMETERS:
110     0302 1
111     0303 1 None.
112     0304 1
113     0305 1 IMPLICIT OUTPUTS:
114     0306 1
115     0307 1 Mount or dismount system service issued.
116     0308 1
117     0309 1 ROUTINE VALUE:
118     0310 1
119     0311 1 1 If successful
120     0312 1 Otherwise : Error status from mount/dismount system service
121     0313 1
122     0314 1 SIDE EFFECTS:
123     0315 1
124     0316 1 None.
125     0317 1
126     0318 1 -
127     0319 1
128     0320 1
129     0321 2 BEGIN           ! Start of CSP$MOUNT
130     0322 2
131     0323 2 LOCAL
132     0324 2
133     0325 2 UIC,
134     0326 2 STATUS,
135     0327 2 BUFFER : REF BLOCK;
136     0328 2
137     0329 2 BUFFER = .CSD [CSD$L_SENDOFF];       ! Get address of message
138     0330 2
139     0331 2 IF ((UIC = .CSD [CSD$L_P1]) NEQ 0)   ! A non-zero P1 is a mount request
140     0332 2 THEN
```

```

141 0333 2
142 0334 2
143 0335 2
144 0336 2
145 0337 2
146 0338 2
147 0339 2
148 0340 2
149 0341 2
150 0342 2
151 0343 2
152 0344 2
153 0345 2
154 0346 2
155 0347 2
156 0348 2
157 0349 2
158 0350 2
159 0351 2
160 0352 2
161 0353 2
162 0354 2
163 0355 2
164 0356 2
165 0357 2
166 0358 2
167 0359 2
168 0360 2
169 0361 2
170 0362 2
171 0363 2
172 0364 2
173 0365 2
174 0366 2
175 0367 2
176 0368 2
177 0369 2
178 0370 2
179 0371 2
180 0372 1

        BEGIN
        LOCAL
          ARG      : VECTOR [2],
          OLD_UIC;
        CSP_MOUNT_DECIPHER (.BUFFER);
        STATUS = CHECK_DEVICE (.BUFFER);
        IF NOT .STATUS
        THEN
          RETURN SSS NORMAL;
          OLD_UIC = $CMKRNL (ROUTIN = GET_UIC);
          ARG [0] = 1;
          ARG [1] = .UIC;
          $CMKRNL (ROUTIN = SET_UIC, ARGLST = ARG);
          STATUS = $MOUNT (ITMLST = .BUFFER);
          ARG [1] = .OLD_UIC;
          $CMKRNL (ROUTIN = SET_UIC, ARGLST = ARG);
        END
        ELSE
        BEGIN
        LOCAL
          DEV_DSC,
          DISM_FLAGS;
        CSP_DISMOUNT_DECIPHER ( .BUFFER, DEV_DSC, DISM_FLAGS ); ! Decipher the cluster-
                                                               ! dismount packet
        STATUS = $DISMOU ( DEVNAM=.DEV_DSC, FLAGS=.DISM_FLAGS ); ! Dismount
        END;
        RETURN .STATUS;
      END ;

```

.TITLE CSPMOUNT
.IDENT \V04-000\

.EXTRN SYSSCMKRNL, SYSSMOUNT
.EXTRN SYSSDISMOU

.PSECT \$CODES,NOWRT,2

| | | | | | | |
|----|----|----|-------|--------------|------------------|--|
| | 3C | BB | 00000 | CSP\$MOUNT:: | | |
| 5E | 10 | C2 | 00002 | PUSHR | #^M<R2,R3,R4,R5> | |
| 53 | 16 | D0 | 00005 | SUBL2 | #16, SP | |
| 52 | 52 | D0 | 00009 | MOVL | 22(CSD), BUFFER | |
| | | 5F | 13 | MOVL | 82(CSD), UIC | |
| | | 53 | DD | BEQL | 2\$ | |
| | | | 0000F | PUSHL | BUFFER | |

CSP\$MOUNT - MOUNT client support for CSP

N 4
16-Sep-1984 01:14:34
14-Sep-1984 13:18:02
VAX-11 Bliss-32 v4.0-742
[SYSLOA.SRC]CSPMOUNT.B32;1Page 5
(2)

| | | |
|--------------|-------------------|---------------------------------|
| 0000V CF | 01 FB 00011 | CALLS #1, CSP_MOUNT_DECIPHER |
| 0000V CF | 53 DD 00016 | PUSHL BUFFER |
| 55 | 01 FB 00018 | CALLS #1, CHECK_DEVICE |
| 05 | 50 DO 0001D | MOVL R0, STATUS |
| 50 | 55 E8 00020 | BLBS STATUS, 1\$ |
| | 01 DO 00023 | MOVL #1, R0 |
| | 64 11 00026 | BRB 4\$ |
| | 7E D4 00028 | CLRL -(SP) |
| 00000000G 00 | 0000V CF 9F 0002A | PUSHAB GET_UIC |
| 08 54 | 02 FB 0002E | CALLS #2, SYSSCMKRN |
| OC AE | 50 DO 00035 | MOVL R0, OLD_UIC |
| | 01 DO 00038 | MOVL #1, ARG |
| | 52 DO 0003C | MOVL UIC, ARG+4 |
| | AE 9F 00040 | PUSHAB ARG |
| 00000000G 00 | 0000V CF 9F 00043 | PUSHAB SET_UIC |
| | 02 FB 00047 | CALLS #2, SYSSCMKRN |
| 00000000G 00 | 53 DD 0004E | PUSHL BUFFER |
| 0C 55 | 01 FB 00050 | CALLS #1, SYSSMOUNT |
| | 50 DO 00057 | MOVL R0, STATUS |
| OC AE | 54 DO 0005A | MOVL OLD_UIC, ARG+4 |
| | AE 9F 0005E | PUSHAB ARG |
| 00000000G 00 | 0000V CF 9F 00061 | PUSHAB SET_UIC |
| | 02 FB 00065 | CALLS #2, SYSSCMKRN |
| | 1B 11 0006C | BRB 3\$ |
| | 5E DD 0006E | PUSHL SP |
| | 08 AE 9F 00070 | PUSHAB DEV_DSC |
| | 53 DD 00073 | PUSHL BUFFER |
| 0000V CF | 03 FB 00075 | CALLS #3, CSP_DISMOUNT_DECIPHER |
| | 6E DD 0007A | PUSHL DISM_FLAGS |
| 00000000G 00 | 08 AE DD 0007C | PUSHL DEV_DSC |
| 55 | 02 FB 0007F | CALLS #2, SYSSDISMOU |
| 50 | 50 DO 00086 | MOVL R0, STATUS |
| 5E | 55 DO 00089 | MOVL STATUS, R0 |
| | 10 CO 0008C | ADDL2 #16, SP |
| | 3C BA 0008F | POPR ^M<R2,R3,R4,R5> |
| | 05 00091 | RSB |

: Routine Size: 146 bytes, Routine Base: \$CODES + 0000

: 181 0373 1

183 0374 1
184 0375 1 ZSBTTL 'CSP_MOUNT_DECIPHER -Deciphers a packet into MOUNT itemlist'
185 0376 1 ROUTINE CSP_MOUNT_DECIPHER (BUFFER) : NOVALUE =
186 0377 1
187 0378 1 !+
188 0379 1
189 0380 1 FUNCTIONAL DESCRIPTION:
190 0381 1
191 0382 1 This routine takes a cluster-mount packet and returns
192 0383 1 an item list.
193 0384 1
194 0385 1 CALLING SEQUENCE:
195 0386 1
196 0387 1 CSP_MOUNT_DECIPHER (ARG1)
197 0388 1
198 0389 1 INPUTS:
199 0390 1
200 0391 1 ARG1 : Address of the input buffer
201 0392 1
202 0393 1 OUTPUTS:
203 0394 1
204 0395 1 None.
205 0396 1
206 0397 1 IMPLICIT INPUTS:
207 0398 1
208 0399 1 None.
209 0400 1
210 0401 1 OUTPUT PARAMETERS:
211 0402 1
212 0403 1 None.
213 0404 1
214 0405 1 IMPLICIT OUTPUTS:
215 0406 1
216 0407 1 None.
217 0408 1
218 0409 1 ROUTINE VALUES:
219 0410 1
220 0411 1 None.
221 0412 1
222 0413 1 SIDE EFFECTS:
223 0414 1
224 0415 1 The cluster-mount packet in the buffer is transformed into
225 0416 1 a mount item list.
226 0417 1
227 0418 1
228 0419 1 NOTES:
229 0420 1
230 0421 1 This decipher routine takes the given cluster-mount packet of the form
231 0422 1 shown below and transforms the packet into an item list.
232 0423 1
233 0424 1
234 0425 1
235 0426 1
236 0427 1
237 0428 1
238 0429 1
239 0430 1

| | Offset |
|-----------------|-------------------------|
| code1 : len1 | 0 ITEM LENG item_desc_1 |
| offset to str_1 | 4 ITEM ADDR |
| unused | 8 ITEM NULL |

CSP_MOUNT_DECIPHER -Deciphers a packet into MOU

0431 1 | +-----+
0432 1 | | code2 | len2 | 0 ITEM_LEN item_desc_2
0433 1 +-----+
0434 1 | offset to str_2 | 4 ITEM_ADDR
0435 1 +-----+
0436 1 | unused | 8 ITEM_NULL
0437 1 +-----+
0438 1 .
0439 1 .
0440 1 .
0441 1 +-----+
0442 1 | 0 | End of item descriptors
0443 1 +-----+
0444 1 | str_1 |
0445 1 +-----+
0446 1 | |
0447 1 +-----+
0448 1 | str_2 |
0449 1 +-----+
0450 1 | |
0451 1 +-----+

1. Each address in the item descriptor is "relocated" to be the
offset from the beginning of the packet (i.e. self-relative).
The transformation is simply to calculate the address in each
item descriptor.

0454 1 | -
0455 1 |
0456 1 |
0457 1 |
0458 1 |
0459 1 | -
0460 1 |
0461 1 |
0462 2 BEGIN ! Start of CSP_MOUNT_DECIPHER
0463 2 |
0464 2 MAP
0465 2 | BUFFER : REF BLOCK [.BYTE];
0466 2 |
0467 2 LOCAL ITEM : REF BLOCK [,BYTE]; ! Pointer to item descriptor
0468 2 |
0469 2 |
0470 2 MACRO ITEM_LEN = 0,0,16,0%; ! Define buffer offsets
0471 2 MACRO ITEM_CODE = 2,0,16,0%;
0472 2 MACRO ITEM_ADDR = 4,0,32,0%;
0473 2 MACRO ITEM_NULL = 8,0,32,0%;
0474 2 LITERAL ITEM_SIZE = 12;
0475 2 |
0476 2 |
0477 2 | ! For each item descriptor, calculate the real address of the item.
0478 2 |
0479 2 |
0480 2 |
0481 2 ITEM = .BUFFER; ! Point to the beginning of buffer
0482 2 WHILE (.ITEM [ITEM_CODE] NEQ 0) DO
0483 2 BEGIN
0484 2 | ITEM [ITEM_ADDR] = .ITEM [ITEM_ADDR] + .BUFFER; ! Calculate the real address
0485 2 | of the item string
0486 2 | ITEM = .ITEM + ITEM_SIZE; ! Bump item descriptor pointer
0487 2 END;

CSPMOUNT
V04-000

D 5
16-Sep-1984 01:14:34 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:18:02 [SYSLOA.SRC]CSPMOUNT.B32;1
Page 8 (3)

: 297 0488 2
: 298 0489 2 RETURN;
: 299 0490 2
: 300 0491 1 END;

! End of CSP_MOUNT_DECIPHER

0000 00000 CSP_MOUNT_DECIPHER:

| | | | | | | | |
|----|----|----|----|-------|-------|--------------|-----------------|
| 50 | 04 | AC | D0 | 00002 | .WORD | Save nothing | 0376 |
| | 02 | A0 | B5 | 00006 | MOVL | BUFFER ITEM | 0481 |
| | | 0A | 13 | 00009 | TSTW | 2(ITEMS) | 0482 |
| 04 | A0 | 04 | AC | C0 | 0000B | BEQL | 2\$ |
| | 50 | | OC | C0 | 00010 | ADDL2 | BUFFER, 4(ITEM) |
| | | | F1 | 11 | 00013 | ADDL2 | #12, ITEM |
| | | | | 04 | 00015 | BRB | 1\$ |
| | | | | | | RET | |
| | | | | | | | |

: Routine Size: 22 bytes. Routine Base: \$CODE\$ + 0092

: 301 0492 1
: 302 0493 1

CSPD
V04-

304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360

0494 1
0495 1 XSBTTL 'CSP_DISMOUNT_DECIPHER -Deciphers a packet into DISMOU arguments'
0496 1 ROUTINE CSP_DISMOUNT_DECIPHER (BUFFER, DEV_DSC, FLAGS) : NOVALUE =
0497 1 +
0498 1
0499 1
0500 1 FUNCTIONAL DESCRIPTION:
0501 1
0502 1 This routine takes a cluster-dismount packet and returns
0503 1 a device descriptor and the dismount flags.
0504 1
0505 1 CALLING SEQUENCE:
0506 1
0507 1
0508 1
0509 1
0510 1
0511 1 ARG1 : Address of the input buffer
0512 1
0513 1 OUTPUTS:
0514 1
0515 1 None.
0516 1
0517 1 IMPLICIT INPUTS:
0518 1
0519 1 None.
0520 1
0521 1
0522 1 OUTPUT PARAMETERS:
0523 1
0524 1 ARG2 : Address of a longword to receive the address
0525 1 of the device descriptor
0526 1 ARG3 : Address of a longword to receive the flags
0527 1
0528 1 IMPLICIT OUTPUTS:
0529 1
0530 1 None.
0531 1
0532 1
0533 1
0534 1
0535 1
0536 1
0537 1
0538 1
0539 1
0540 1
0541 1
0542 1
0543 1
0544 1
0545 1
0546 1
0547 1
0548 1
0549 1
0550 1
NOTES:
This decipher routine takes the given cluster-dismount packet of the form
shown below and returns a device descriptor and the dismount flags.

| | Offset |
|----------------|-------------|
| flags | 0 BUF_FLAGS |
| dev descriptor | 4 BUF_DSC |

```

361 0551 1 | | 8
362 0552 1 | | |
363 0553 1 | | device string | 12 BUF_STR
364 0554 1 | | |
365 0555 1 | | |
366 0556 1 | | |
367 0557 1 | | |
368 0558 1 | | !-
369 0559 1 | | |
370 0560 2 BEGIN ! Start of CSP_DISMOUNT_DECIPHER
371 0561 2 |
372 0562 2 MAP
373 0563 2 BUFFER : REF BLOCK [.BYTE] ;
374 0564 2 |
375 0565 2 LOCAL LOC_DSC : REF BLOCK [.BYTE] ;
376 0566 2 |
377 0567 2 |
378 0568 2 |
379 0569 2 MACRO BUF_FLAG = 0,0,32,0%; ! Define buffer offsets
380 0570 2 MACRO BUF_DSC = 4,0,32,0%; !
381 0571 2 MACRO BUF_STR = 12,0,32,0%; !
382 0572 2 LITERAL BUF_HDR_LEN = 12;
383 0573 2 |
384 0574 2 .FLAGS = .BUFFER[BUF_FLAG]; ! Get flags from buffer
385 0575 2 LOC_DSC = BUFFER[BUF_DSC]; ! Point to device descriptor
386 0576 2 LOC_DSC[DSCSA_POINTER] = .LOC_DSC[DSCSA_POINTER] + .BUFFER; ! 'Relocate' address
387 0577 2 | in device descriptor
388 0578 2 .DEV_DSC = .LOC_DSC; ! Return address of device dsc
389 0579 2 |
390 0580 2 RETURN;
391 0581 1 END; ! End of CSP_DISMOUNT_DECIPHER

```

| 0000 00000 CSP_DISMOUNT DECIPHER: | | | | | | | | | |
|-----------------------------------|-------|----|-------|----------|-------|---------------------|--|--|--|
| 50 | 0C BC | 04 | BC D0 | 00002 | .WORD | Save nothing | | | |
| | 04 AC | 04 | C1 D0 | 00007 | MOVL | ABUFFER, AFLAGS | | | |
| | 04 AO | 04 | AC C0 | 0000C | ADDL3 | #4, BUFFER, LOC_DSC | | | |
| | 08 BC | 50 | D0 D0 | 00011 | ADDL2 | BUFFER, 4(LOC_DSC) | | | |
| | | | | 04 00015 | MOVL | LOC_DSC, ADEV_DSC | | | |
| | | | | | RET | | | | |

; Routine Size: 22 bytes, Routine Base: SCODES + 00AB

: 392 0582 1

```

0583 1 XSBTTL 'GET_UIC'           - Get our process UIC'
0584 1 ROUTINE GET_UIC =
0585 1 ++
0586 1
0587 1 FUNCTIONAL DESCRIPTION:
0588 1
0589 1 This is a kernel-mode routine to get the UIC of a process.
0590 1
0591 1 CALLING SEQUENCE:
0592 1
0593 1     GET_UIC ()
0594 1
0595 1 INPUT PARAMETERS:
0596 1
0597 1     None.
0598 1
0599 1 IMPLICIT INPUTS:
0600 1
0601 1     None.
0602 1
0603 1 OUTPUT PARAMETERS:
0604 1
0605 1     None.
0606 1
0607 1 IMPLICIT OUTPUTS:
0608 1
0609 1     None.
0610 1
0611 1 ROUTINE VALUE:
0612 1
0613 1     UIC of this process.
0614 1
0615 1 SIDE EFFECTS:
0616 1
0617 1     None.
0618 1
0619 1 !--
0620 1
0621 1 BEGIN
0622 1
0623 2 EXTERNAL
0624 2     SCH$GL_CURPCB : REF BLOCK [, BYTE] ADDRESSING_MODE (ABSOLUTE);
0625 2             ! system address of process PCB
0626 2
0627 2 RETURN (.SCH$GL_CURPCB[PCBSL_UIC]);
0628 2
0629 2 END;                                ! End of routine GET_UIC

```

.EXTRN SCH\$GL_CURPCB

| | | | | |
|--------------|----------------|----------------|---------------------|--------|
| 50 00000000G | 9F 00 00000000 | GET_UIC: .WORD | Save nothing | : 0585 |
| 50 00BC | C0 00 00000009 | MOVL | @SCH\$GL_CURPCB, R0 | : 0629 |
| | | MOVL | 188(R0), R0 | |

CSPMOUNT
V04-000

GET_UIC - Get our process UIC

H 5
16-Sep-1984 01:14:34
14-Sep-1984 13:18:02

VAX-11 Bliss-32 v4.0-742
[SYSLOA.SRC]CSPMOUNT.B32;1

Page 12
(5)

; 0631

; Routine Size: 15 bytes, Routine Base: \$CODES + 00BE

; 443 0632 1

```

445      0633 1
446      0634 1 %SBTTL 'SET_UIC'           = Set our process UIC'
447      0635 1 ROUTINE SET_UIC ( UIC ) =
448      0636 1
449      0637 1 ++
450      0638 1
451      0639 1 FUNCTIONAL DESCRIPTION:
452      0640 1
453      0641 1 This is a kernel-mode routine to set the UIC of a process.
454      0642 1
455      0643 1 CALLING SEQUENCE:
456      0644 1
457      0645 1     SET_UIC (ARG1)
458      0646 1
459      0647 1 INPUT PARAMETERS:
460      0648 1
461      0649 1     ARG1    : Desired UIC
462      0650 1
463      0651 1 IMPLICIT INPUTS:
464      0652 1
465      0653 1     None.
466      0654 1
467      0655 1 OUTPUT PARAMETERS:
468      0656 1
469      0657 1     None.
470      0658 1
471      0659 1 IMPLICIT OUTPUTS:
472      0660 1
473      0661 1     None.
474      0662 1
475      0663 1 ROUTINE VALUE:
476      0664 1
477      0665 1     1.
478      0666 1
479      0667 1 SIDE EFFECTS:
480      0668 1
481      0669 1     None.
482      0670 1
483      0671 1 --
484      0672 1
485      0673 2 BEGIN
486      0674 2
487      0675 2 EXTERNAL
488      0676 2     SCH$GL_CURPCB : REF BLOCK [, BYTE] ADDRESSING MODE (ABSOLUTE):
489      0677 2                                     ! System address of process PCB
490      0678 2     SCH$GL_CURPCB [PCBSL_UIC] = .UIC;          ! Set UIC
491      0679 2
492      0680 2 RETURN 1;
493      0681 2
494      0682 2 END;                                ! End of routine SET_UIC

```

50 00000000G 9F 0000 00000 SET_UIC:.WORD MOVL Save nothing
 @SCH\$GL_CURPCB, R0

: 0635
 : 0678

CSPMOUNT
V04-000

SET_UIC - Set our process UIC

16-Sep-1984 01:14:34
14-Sep-1984 13:18:02
VAX-11 Bliss-32 v4.0-742
[SYSLOA.SRC]CSPMOUNT.B32;1

Page 14
(6)

00BC C0 04 AC 00 00009
50 01 00 0000F
04 00012
MOVL
MOVL
RET

; 0680
; 0682

; Routine Size: 19 bytes, Routine Base: SCODES + 00CD

; 495 0683 1
; 496 0684 1

CSPC
V04-

SET_UIC - Set our process UIC

498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554

0685 1
0686 1 XSBTTL 'CHECK_DEVICE' - Check if the mount request should be processed'
0687 1 ROUTINE CHECK_DEVICE (BUFFER) =
0688 1 +
0689 1
0690 1
0691 1 FUNCTIONAL DESCRIPTION:
0692 1
0693 1 This routine determines if the mount request received should
0694 1 be processed. If the target device is already mounted, or is
0695 1 not a cluster-wide device, then the request should be dismissed.
0696 1
0697 1 CALLING SEQUENCE:
0698 1
0699 1 CHECK_DEVICE (ARG1)
0700 1
0701 1 INPUTS:
0702 1
0703 1 ARG1 : Address of the mount item list
0704 1
0705 1 OUTPUTS:
0706 1
0707 1 None.
0708 1
0709 1 IMPLICIT INPUTS:
0710 1
0711 1 None.
0712 1
0713 1 OUTPUT PARAMETERS:
0714 1
0715 1 None.
0716 1
0717 1 IMPLICIT OUTPUTS:
0718 1
0719 1 None.
0720 1
0721 1 ROUTINE VALUES:
0722 1
0723 1 0 : If the mount request should be dismissed.
0724 1 1 : If the mount request should be processed.
0725 1
0726 1
0727 1
0728 1
0729 1
0730 1
0731 1
0732 1
0733 1
0734 1
0735 1
0736 1
0737 1
0738 1
0739 1
0740 1
0741 1
SIDE EFFECTS:
None.
-
BEGIN ! Start of CHECK_DEVICE
MAP
LOCAL BUFFER : REF BLOCK [,BYTE];
LOCAL STATUS,
LOCAL_EFN,
ITEM : REF BLOCK [,BYTE], : Local event flag
ITEM_DESCRIPTOR : Pointer to item descriptor

```

CHECK_DEVICE - Check if the mount request shou 16-Sep-1984 01:14:34 VAX-11 Bliss-32 v4.0-742
                                                14-Sep-1984 13:18:02 [SYSLOA.SRC]CSPMOUNT.B32;1

555 0742 2 DEV_DESC : BLOCK [DSC$K_S_BLN, BYTE], ! Target device descriptor
556 0743 2 DEVCHAR : BLOCK [4, BYTE], ! Device char word buffer
557 0744 2 DEVCHAR2 : BLOCK [4, BYTE], ! 2nd device char word buffer
558 0745 2 ITMLST : BLOCK [(2*12)+4, BYTE] INITIAL
559 0746 2
560 0747 2
561 0748 2
562 0749 2
563 0750 2
564 0751 2
565 0752 2
566 0753 2
567 0754 2
568 0755 2
569 0756 2
570 0757 2
571 0758 2
572 0759 2
573 0760 2
574 0761 2
575 0762 2 EXTERNAL ROUTINE
576 0763 2 LIB$GET_EF : ADDRESSING_MODE (GENERAL), ! RTL routine to get an EF
577 0764 2 LIB$FREE_EF : ADDRESSING_MODE (GENERAL); ! RTL routine to release the EF
578 0765 2
579 0766 2 MACRO ITEM_LEN = 0,0,16,0%; ! Define buffer offsets
580 0767 2 MACRO ITEM_CODE = 2,0,16,0%
581 0768 2 MACRO ITEM_ADDR = 4,0,32,0%
582 0769 2 MACRO ITEM_NULL = 8,0,32,0%
583 0770 2 LITERAL ITEM_SIZE = 12;
584 0771 2
585 0772 2 STATUS = 0; ! Assume failure
586 0773 2 ITEM = .BUFFER; ! Point to the beginning of buffer
587 0774 2 LIB$GET_EF (LOCAL_EFN); ! Get a local event flag
588 0775 2
589 0776 2
590 0777 2 ! Scan the item list for device names. For each device name in item list,
591 0778 2 issue a $GETDVI system service to find out the status of the device.
592 0779 2
593 0780 2 WHILE (.ITEM [ITEM_CODE] NEQ 0 ) DO ! Examine each item
594 0781 3 BEGIN
595 0782 3 IF .ITEM [ITEM_CODE] EQL MNT$_DEVNAM
596 0783 3 THEN
597 0784 4 BEGIN ! For device names only
598 0785 4 DEV_DESC [DSC$B_DTYPE] = 0; ! Set up device descriptor
599 0786 4 DEV_DESC [DSC$B_CLASS] = 0;
600 0787 4 DEV_DESC [DSC$W_LENGTH] = .ITEM [ITEM_LEN];
601 0788 4 DEV_DESC [DSC$A_POINTER] = .ITEM [ITEM_ADDR];
602 0789 4
603 P 0790 4 STATUS = $GETDVIW ( DEVNAM = DEV_DESC, ! Get device info
604 P 0791 4 ITMLST = ITMLST,
605 0792 4 EFN = .LOCAL_EFN );
606 0793 4
607 0794 5 IF ( NOT .STATUS ) ! If $GETDVI failed
608 0795 5 OR ( .DEVCHAR [DEVS$V_MNT] ) ! or device already mounted
609 0796 5 OR ( NOT .DEVCHAR2 [DEVS$V_CLU] ) ! or not cluster-wide device
610 0797 4 THEN
611 0798 5 BEGIN

```

```

: 612 0799 5 STATUS = 0;
: 613 0800 5 EXITLOOP;
: 614 0801 4 END;
: 615 0802 3 ITEM = .ITEM + ITEM_SIZE;
: 616 0803 3 END;
: 617 0804 2 ! Bump item descriptor pointer
: 618 0805 2 ! End of while loop
: 619 0806 2 LIB$FREE_EF (LOCAL_EFN);
: 620 0807 2 ! Release the event flag
: 621 0808 2 ! Back to caller
: 622 0809 2 ! End of CHECK_DEVICE
: 623 0810 1 END;

```

.PSECT \$SPLIT\$,NOWRT,NOEXE,2

| | | | | |
|----------|-------|--------|-------|-----|
| 0004 | 00000 | P.AAA: | .WORD | 4 |
| 0002 | 00002 | | .WORD | 2 |
| 00000000 | 00004 | | .LONG | 0 |
| 00000000 | 00008 | | .LONG | 0 |
| 0004 | 0000C | | .WORD | 4 |
| 00E6 | 0000E | | .WORD | 230 |
| 00000000 | 00010 | | .LONG | 0 |
| 00000000 | 00014 | | .LONG | 0 |
| 00000000 | 00018 | | .LONG | 0 |

.EXTRN LIB\$GET_EF, LIB\$FREE_EF
.EXTRN SYSSGETDVIW

.PSECT \$CODE\$,NOWRT,2

003C 00000 CHECK_DEVICE:

| | | | | |
|----------------|----------------|--------|---------------------|--------|
| OC AE 0000' 5E | 30 C2 00002 | .WORD | Save R2,R3,R4,R5 | : 0687 |
| 10 AE | 1C 28 00005 | SUBL2 | #48, SP | : 0760 |
| 1C AE | 6E 9E 0000C | MOVC3 | #28, P.AAA, ITMLST | : 0733 |
| | AE 9E 00010 | MOVAB | DEVCHAR, ITMLST+4 | |
| | 53 D4 00015 | MOVAB | DEVCHAR2, ITMLST+16 | |
| 52 | AC D0 00017 | CLRL | STATUS | : 0772 |
| | 08 AE 9F 0001B | MOVL | BUFFER, ITEM | : 0773 |
| 00000000G 00 | 01 FB 0001E | PUSHAB | LOCAL_EFN | : 0774 |
| | 02 A2 B5 00025 | CALLS | #1, LIB\$GET_EF | |
| | 3D 13 00028 | TSTW | 2(ITEM) | : 0780 |
| | 02 A2 B1 0002A | BEQL | 4S | |
| | 32 12 0002E | CMPW | 2(ITEM), #1 | : 0782 |
| 28 AE | 62 3C 00030 | BNEQ | 3S | |
| 2C AE | A2 D0 00034 | MOVZWL | (ITEM), DEV_DESC | : 0787 |
| | 7E 7C 00039 | MOVL | 4(ITEM), DEV_DESC+4 | : 0788 |
| | 7E 7C 0003B | CLRQ | -(SP) | : 0792 |
| | 1C AE 9F 0003D | CLRQ | -(SP) | |
| | 3C AE 9F 00040 | PUSHAB | ITMLST | |
| | 7E D4 00043 | PUSHAB | DEV_DESC | |
| | AE DD 00045 | CLRL | -(SP) | |
| 00000000G 00 | 08 FB 00048 | PUSHL | LOCAL_EFN | |
| 53 | 50 D0 0004F | CALLS | #8, SYSSGETDVIW | |
| 09 | 53 E9 00052 | MOVL | R0, STATUS | |
| | | BLBC | STATUS, 28 | : 0794 |

CSPMOUNT
V04-000

N 5
16-Sep-1984 01:14:34 14-Sep-1984 13:18:02 VAX-11 Bliss-32 V4.0-742
[SYSLOA.SRC]CSPMOUNT.B32;1

| | | | | | | | | | |
|----|-----------|----|----|----|-------|-------|--------|--------------------|--------|
| 04 | 02 | AE | 04 | 03 | E0 | 00055 | BBS | #3, DEVCHAR+2, 2\$ | : 0795 |
| | | | | AF | E8 | 0005A | BLBS | DEVCHAR2, 3\$ | : 0796 |
| | | | | 53 | D4 | 0005E | CLRL | STATUS | : 0799 |
| | | | | 05 | 11 | 00060 | BRB | 4S | : 0798 |
| | | | 52 | OC | C0 | 00062 | ADDL2 | #12, ITEM | : 0803 |
| | | | | BE | 11 | 00065 | BRB | 1S | : 0780 |
| | 00000000G | 00 | 08 | AE | 9F | 00067 | PUSHAB | LOCAL_EFN | : 0806 |
| | | | | 01 | FB | 0006A | CALLS | #1, LIBSFREE_EF | : 0808 |
| | | | 50 | 53 | D0 | 00071 | MOVL | STATUS, R0 | : 0810 |
| | | | | 04 | 00074 | | RET | | |

: Routine Size: 117 bytes, Routine Base: \$CODE\$ + 00E0

: 624 0811 1
: 625 0812 1 END
: 626 0813 0 ELUDOM

: ! End of CSPMOUNT

PSECT SUMMARY

| Name | Bytes | Attributes |
|----------|---|------------|
| \$CODE\$ | 341 NOVEC,NOWRT, RD : EXE,NOSHR, LCL: REL: CON,NOPIC,ALIGN(2) | |
| \$SPLITS | 28 NOVEC,NOWRT, RD :NOEXE,NOSHR, LCL: REL: CON,NOPIC,ALIGN(2) | |

Library Statistics

| File | ----- | Symbols | ----- | Pages | Processing |
|-----------------------------------|-------|---------|---------|--------|------------|
| | Total | Loaded | Percent | Mapped | Time |
| \$_\$255\$DUA2B:[SYSLIB]LIB.L32;1 | 18619 | 18 | 0 | 1000 | 00:01.4 |

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:CSPMOUNT/OBJ=OBJ\$:CSPMOUNT MSRC\$:CSPMOUNT/UPDATE=(ENH\$:CSPMOUNT)

: Size: 341 code + 28 data bytes
: Run Time: 00:08.6
: Elapsed Time: 00:39.7
: Lines/CPU Min: 5645
: Lexemes/CPU-Min: 29986
: Memory Used: 109 pages
: Compilation Complete

CSP
V04-

57
2C
20
6F
74
49
20
20
57
20
20
73
49
6F
69
45
51
61
45
6F
61
45
6F
61
45
73
45
6F
61
45
75
72
45
51
24
65

45
6F
61
45
6F
61
45
75
72
45
51
24
65

0394 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

